An Open-Source, Low-Cost Apparatus for Conductivity Measurements Based on Arduino and Coupled to a Handmade Cell

Giovanni Visco, Emanuele Dell'Aglio *, Mauro Tomassetti, Luca Ugo Fontanella and Maria Pia Sammartino

Department of Chemistry, La Sapienza University of Rome, 00185 Rome, Italy; giovanni.visco@fondazione.uniroma1.it (G.V.); mauro.tomassetti@uniroma1.it (M.T.); lucaugo.fontanella@uniroma1.it (L.U.F.); mariapia.sammartino@uniroma1.it (M.P.S.)
* Correspondence: emanuele.dell.aglio@gmail.com

How to build it, step by step

Disclaimer

The authors will therefore not be responsible for any misinterpretation or misuse of the information provided in the article and this guide. As with many other do-it-yourself projects, in any case but especially if carried out by people unfamiliar with the tools, the process can be dangerous, so replication of the prototype is "at your own risk" and the authors will not be responsible for damage not only to the experimenter but to any other person or property.

We make no representations or warranties of any kind, express or implied, written or oral, statutory or otherwise, with respect to the information, including but not limited to its condition, quality, performance, merchantability or fitness for purpose.

Use of the device produced by this DIY (Do It Yourself) guide in life support and/or safety applications is strictly prohibited and entirely at the investigator's own risk.

Introduction

The construction of the electrode and the interface circuit, which we could call Shield in the Arduino jargon is within the reach of any teaching laboratory of a high school, even easier to build the construction in an electrochemical university laboratory.

With this guide we want to describe step by step the construction of the conductivity measuring instrument that uses Arduino as a data acquisition system as well as power supply.

It is not the task of this guide to explain the operation of Arduino itself and its IDE (Integrated Development Environment) for its programming, even less means a guide to C or C ++ language.

This text can be used to buy the electronic components, to produce the necessary PCBs, to install and weld the components in the 2 PCBs in the right order, to test the electrode and the shield just mounted and eventually solve problems and finally to install the software, free, supplied with this guide. A calibration procedure is also described for whose full description refers to the original article.

If you have never used Arduino, his shields, his IDE is advised to read a very simple book, {1} and then deepen with the book of one of the founders of the project {2}.

First step, the PCB

2 printed circuit boards (PCBs) are required to build this measuring instrument. They can be made starting from the electrical circuit described below..

To simplify the construction together with the original article, in Supplementary Material (SM) there are 2 .ZIP files with the Gerber (Extended Gerber RS274X Version) files. The electrical circuit was designed using the FreePCB software in its latest version available in 2022; screenshots of the two PCBs are shown in Fig.A.

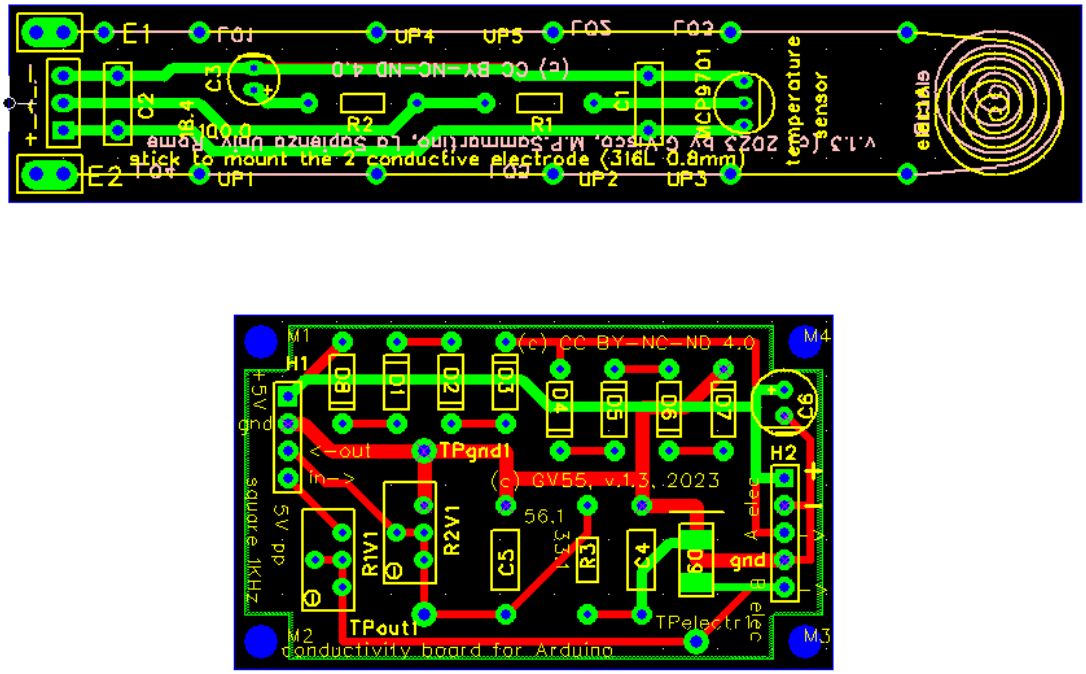


Fig.A, Screenshots of the 2 PCB designed by authors using the FreePCB software

Many DIY projects involve building the basic printed circuit boards at home, which is not recommended here. Instead, we strongly recommend the use of one of the hundreds of PCB-farms that also produce prototypes for only 10 euro. The attached Gerber files (*the-shield-PCB.ZIP* and *the-electrode-PCB.ZIP*) were sent to one of these "farms" and the two resulting PCBs are shown in Fig. B.

PCBs are not complex to manufacture; in order to keep a low cost, the following suggestions can be followed for their construction: FR-4, TG130-140, 2 layers, 6/6 mil minimum track spacing, Solder Mask green, Silk Mask white, HASL finish, Cu 1oz thickness, standard font for Silk Mask.

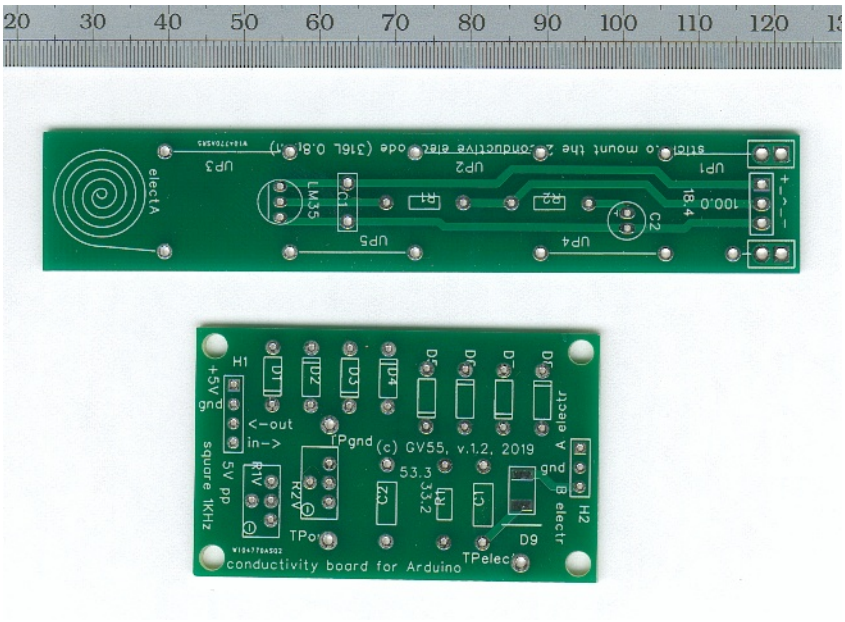


Fig.B, the 2 PCB produced by the PCB farm cited in article

The Bill Of Materials (BOM)

The assembly of the instrument (electrode and shields) requires the choice of the necessary electronic and all the other material to be purchased. Particular attention must be paid to diodes, the 1N4448 and BAT54 are a good choice due to their low forward voltage, on the contrary the 1N4148 is not recommended.

Parts, components, equipment are described here with codes found in the catalogues of major electronic component vendors, they are only a reference, in most of the cases we have added an estimated cost, as an example (Fairchild, 512-1N4448, 0.085 Euro).

Part numbers such as D1, C2, R3, for example, refer to the codes on the 2 PCBs, see fig.A corresponding to the last prototype built.

List of components needed for the shield:
a) H1, Header pin strip, angled, 4pin (Harwin, 24pin, M20-9712446, 0.593 Euro)
b) D1-D8, 1N4448, 8x (Fairchild, 512-1N4448, 0.085 Euro)
c) R1V, trimmer 2K2, type T93, 1x (Vishay, T93XA222KT20, 1.17 Euro)
d) R2V, trimmer 2M0, type T93, 1x (Vishay, T93YA205KT20, 1.27 Euro)
e) C1, 15nF, 1x (TDK, FG28X7R1H153KNT00, 0.161 Euro)
f) C2, 300nF, 1x (TDK, FG24X7R1H334KNT06, 0.237 Euro)
g) R2, 220ohm, 1x (Vishay, CCF07220RJKE36, 0.085 Euro)
h) D9, BAT54 Schottky Diode, 1x (Vishay, BAT54W-HG3-08, 0.313 Euro)
i) C3, 10uF.16V, 1x (Whurt, 860010372001, 0.085 Euro)
j) H2, Header pin strip, 5pin (see point a)

List of components needed for the electrode:
k) Q1, MCP9701A, 1x (Microchip, MCP9701A-E/TO, 0.263 Euro)
l) C1, 0.1 uF, 50V, 1x (Vishay/BC, K104K15X7RF53H5, 0.093 Euro)
m) R1, 1000ohm, 1x (Vishay, SFR25H0001001JA500, 0.085 Euro)
n) C2, 4.7uF, 25V, 1x (Whurt, 860020472001, 0.085 Euro)
o) R2, optional to correct cable impedance, often 50 or 75ohm

p) C3, optional to correct cable impedance, often some pF
q) H3, Header pin strip, 3pin (the same as in point a)
r) strip of Terminal block for 1+1 pin ending of E1, E2, (CUI Devices, 4pin, TB001-500-04BE, 0.635 Euro)
s) electrodes, wire, 0.8mm, 300mm, stainless steel, type 316L, (Langley, LSL-DIY-540, about 200m, 10.6 Euro)
t) epoxy resin for jewellery making, type A:B=3:1, about 3 mL used, (AB Crystal, 100mL, 3.0 Euro)

Arduino and components for its connections:
u) Arduino UNO R3 (Arcuino cc, A000066, 18.63 Euro)
v) H1, Header pin strip, 4pin (Harwin, 24pin, M20-9772446, 0.593 Euro)
w) wire, shielded wire, 1m, RG174 type or similar

The following equipment is required for the assembly and preliminary testing of the circuits; brands and codes are given as examples:
x) thermostat welder up to 40W max, or Solder Station similar to Weller WTCP (Apex, WE1010NA)
y) power supply (PSU) from 5V, better if adjustable and short circuits protected (Extech, 382200)
z) Digital multimeter (DMM) for ohm measurement (Extech, MN10)
aa) a small equipment like pliers with pointed beak, electronic cutters (Phoenix, 1212487), cut and cross screwdrivers
ab) a mercury thermometer with a resolution 0.1 ºC, or better, for measurements between 0 and 50 degrees (ASTM 63C, ASTM 64C), partial immersion type
ac) a portable conductivimeter (Hanna HI70031p) or a lab one (see main article) to be used for reference measures.
ad) at least 2 conductivity standards, for example 84uS (XS Basic, 51100623) and 1430uS (XS Basic, 51100633)
ae) 5 litres of distilled water (conductivity $\leq$5µS) to be used for the production of ice cubes from which the Temperature calibration will start
af) a low power heating cooker (100W or similar) to be used to increase the Temperature of the ice cubes during the Temperature calibration
ag) 2L glass or metal container carefully clean for the calibration of the temperature sensor

The Circuit(s)

The complete circuit is shown in Fig.5 of the article while the following Fig.C and D schematize the electrode and shield circuits respectively. Fig. C only shows the stainless steel spirals, the IC with the power supply circuit and the impedance filter for the measurement of the temperature. In Fig. D the Shield circuit is draft where the members' acronyms correspond to those of the BOM and of the last built PCB.

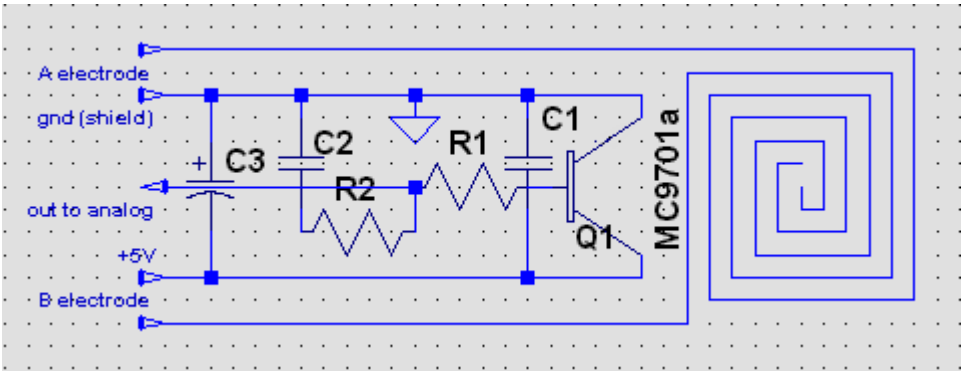


Fig.C the schematic for the electrodes construction including the temperature sensor

The complete description of use/function of every components is present in the file *Arduino-cond-temp-the-circuits.doc* also in SM.

Soldering

A welding control unit with temperature control and a cone tip with a final diameter of 0.8 mm is recommended for assembly. This cannot be a course in soldering, electronics, measurements or programming for Arduino, so some basics on the topics are taken for granted.

Build the electrode

For the assembly of the instrument it is advisable to start from the electrode, you have to start by deciding the size of the stainless steel spirals from which the instrument's response depends. The suggested value is 8 mm in diameter that can be increased up to 15 mm for measurements in low concentration solutions of salts.

The spirals should be as narrow as possible but not overlapping. To make them, proceed as follows
1) Roll a 316L stainless steel wire to a diameter of 0.8 mm and leave at least 100 mm of straight wire.
2) fix a couple of Terminals Block on E1 and E2 (see Fig A)
3) place the two electrodes on the opposite sides of the PCB. For each one, pass the stem through the holes provided until it reaches E1 and E2 respectively (see fig. 3 in the article).
4) tighten the wire in the Terminal Block and weld the pin to the PCB in E1 and E2
5) weld the resistor R1 and the condenser C1
6) weld the C2 capacitor paying attention to the position of the positive pole
7) weld the temperature measurement chip, even here, paying attention to its positioning following the shape of the drawing on the PCB, it should be left up for about 5/7 mm from the PCB

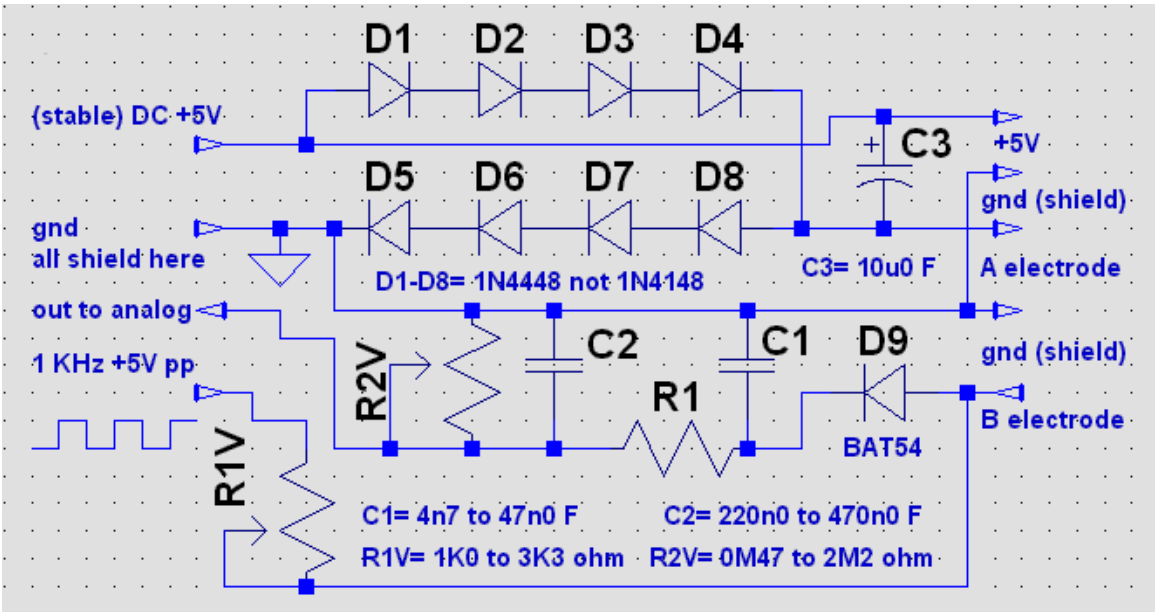


Fig.D schematic for the Shield with possible components values

8) now you can weld the 3-pin terminal (H3) by cutting it from the strip of 24
9) now you have to weld the positive wire to the corresponding pin
10) the central wire of the RG174 is now welded to the central pin named "<"
11) the RG174 shield will be welded to the negative pin, this cable is longer and ends directly in Arduino not to the shield
12) it would be better to use the thermoshrink insulation for the central wire of the RG174. Use a diameter of just higher than the cable in use, about 20mm long and then heated gently with a hair dryer
13) 2 other wires RG174 or similar must be peeled and welded to the tip
14) in the Terminals blocks you must fix with the screws 2 wire of solid copper wire from 1 or 1.2 mm diameter, 10mm long
15) now you can weld the control panel of the 2 shielded wire to the wire piece fixed to the terminals block, holding it the shortest possible.

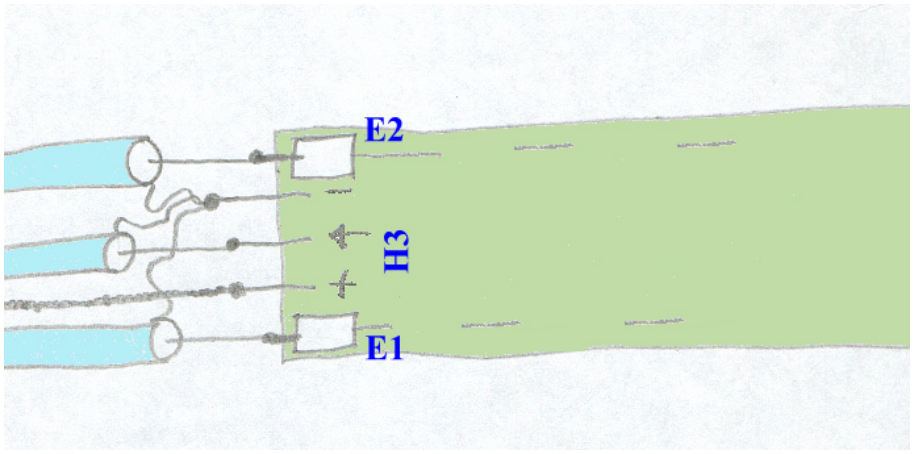


Fig.E drawing of the suggested connection with shielded wire

Be careful that the shielded cables have the central wire very thin and it is very easy to damage it during the peeling and consequently make it prone to break the first fold or movement near the welding. In fig.E a drawing of connections side using shielded cables

A check of the electrodes is suggested before the PCB covering with the resin.
16) connect a PSU, set at 5V, to the two wires with which the electrodes are connected (wires to E1 and E2 terminal block, see fig. A),
17) Measure the voltage on the 2 spirals, the same tension, 5V, must be read
18) disconnect the PSU
19) connect the PSU, always fixed to 5V, to the other end of the positive and negative wires welded to H3
20) measure with the DMM the voltage output of PSU and that on the two side pins of the MC9701a, the same voltage should be read, leave PSU connected
21) measure the voltage between the inner wire of the shielded cable (connected to the center pin of H3) and the ground with the DMM
22) heat the temperature sensor by an hair dryer at a distance of about 200mm, a voltage variation must occur on temperature increase. On turn off the hair dryer, the inverse voltage variation must be observed
23) now you can cut all the excess wires and the reopores with a satin cutter
24) the MC9701a should now be bent towards spirals

The part a little more difficult is to isolate the whole circuit with layers and layers of two-component resin without, absolutely, covering the spirals but covering the temperature sensor and all the other components on PCB that must be accurately isolated from hostile, adverse solution to be measured. At this aim the two spirals have to be protected but with a materials easy to be removed after the resin application.

Prepare a sugar-molasses solution by dissolving 40 grams in 20 ml of warm water. Completely immerse the spiral in the solution while shaking gently; then remove it from the sugar solution and allow it to dry. Repeat the immersion process at least 2 or 3 times, making sure that the sugar layers well adhere to the coil and the underlying PCB.
26) Prepare the epoxy resin by mixing the two components in the ratio suggested by the manufacturer. Gently stir the resulting resin for some minutes.
27) using a spring, pliers or clothes peg, fix the PCB perfectly horizontally and drip the resin over the entire surface, avoiding the spiral (even if protected by the sugar)
28) wait the time suggested by the manufacturer for the resin to harden
29) repeat the operations in points 27 and 28 on the other side of the PCB
30) check the surface carefully and, better still, apply another layer of resin on both sides of the PCB
31) when the resin has hardened proceed to the sugar solubilization in order to uncover the spirals.

Build the shield

The only difficulty in the assembly of the Shield is to comply with the polarities of the individual diodes, of C3 and external connections.

Follow the assembly step by step:
32) clamp with a thin pointed pliers and with an elastic D9 diode paying attention to the polarity marked with a row in the PCB (even the diode has a corresponding line)
33) weld the diode D9 to the circuit
34) welding one to one the diodes from D1 to D8, paying close attention to the rows on the PCB indicating the polarity, also leave these diodes raised by 2mm
35) mount the R2V trimmer, be careful to match the adjustment screw with the PCB design
36) mount the R1V trimmer, be careful to match the adjustment screw with the PCB design
37) before continuing it is necessary to adjust the 2 trimmer for the suggested ohm values
38) with the DMM measures in Ohm the resistance between "TPelectr" and "in->" and adjust R1V for 1000ohm
39) with the DMM measure the resistance between "TPgnd" and "<-out", set by adjusting the R2V screw for 1Mohm
40) mount the 2 C1 and C2 capacitors, which have no polarity to be respected
41) welding the R1 resistor leaving it raised from the 2 mm circuit
42) mount the C3, the PCB strips indicate the negative
43) starting from the side to the electrode, mount the H2 terminal by cutting 5 pins from a) strip
44) the positive wire coming from the electrode must be connected to the "+" pin
45) to the terminals "->" and "->" the central wires of the shielded cables coming from the Terminal Block must be connected
46) the shields, socks, of the two RG174 cables must be soldered one to the "gnd" pin and the other to the "-" (the shielded cable that comes from the temperature sensor "<" goes directly to Arduino)

47) side towards Arduino, mount the H1 header, angled outward
48) the RG174 center wire must be connected to the "in->" pin
49) another RG174 cable should be peeled and the center wire soldered to the "<-out" pin. the shields of this 2 cables must be connected to the "gnd" pin (all the negative shield and electrode power supply passes for the shield of these cables, make excellent welds)
50) connect the appropriate colour wires to "+ 5V"
51) you may not even mount any of the headers, for the electrode and shield, and directly weld the wires on the PCB paying attention to the "gnd" et "-" that connect more wires

Before isolating the Shield better check the operation.
52) connect the PSU to the two wires "+ 5V" and "gnd" and feed the Shield with 5V
53) measure the V value on the "gnd" and "A ->" output pins, 2.5V should be read, measure between “+” and “gnd”, 5V must be obtained.

To isolate the Shield you can use beeswax or commercial paraffin warming it in a bain-marie to no longer 80 ºC and immersing the shield quickly. If necessary repeat when wax is solid.

Build the Arduino connection

In many projects you can see the connections with Arduino headers made with unipolar wires, but it is not a safe connection method. We suggest the one described in v) header from at least 6 pins that cover not only the pins in use but even at least a couple before and after. Even better to use a continuous strip that covers all the pins and weld on it only the necessary wires, as shown in Fig.F of another our project.

Apart the power supply and the Gnd pins all other connections depend on the software, i.e. from the decisions of the programmer, for example which of the 6 analog input pins should be used.

54) the wire RG174 must be welded to pin 5 of Arduino which supply the square-wave signal to the shield, position "in->"
55) the screen of this cable must be welded to the GND pin (immediately after PIN 13) on the same side of Arduino
56) the wire go to shield "+ 5V" pin must welded to the 5V Arduino pin
57) the shielded wire comes from the Shield, position "<-out" must be welded at pin 14, also called A0, of Arduino
58) the shield of this cable must be welded to the second GND (near Vin) from the same side
59) the shielded wire comes directly from the electrode, position "<" must be welded at pin 16, also called A2,
60) its shield is welded to the first GND from the same side

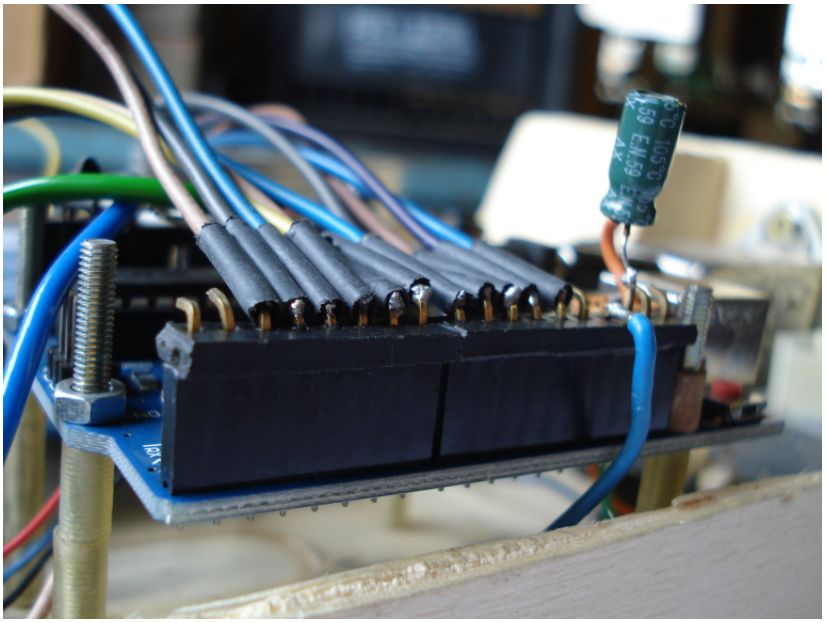


Fig.F, using a single strip the connection are more secure

All this attention to GND and shielded cables is to allow the weak mV signals from the temperature sensor and electrodes to be carried to the Arduino even at a distance of a few metres..

Install the Arduino software

Before installing the management firmware for this instrument, it is necessary to install the IDE on a PC, downloading it from the Arduino.cc site. The firmware has been written with version 1.8.9 for Windows, compatible also with other versions up to the time of writing this text.

With the help of one of the recommended books, try compiling "Examples, 01 Basics. Blink "and check that the Arduino LED flashes indicating that the Arduino drivers and connections are working.

After all the previous steps for building and connections have been done, the software must be compiled and installed via upload.

61) in the IDE of Arduino with file-open load the software, present in the SM of the paper, "*Conduct-temp1.2.ino*", and with sketch-upload upload the firmware in the Arduino board.
62) open Tools-Serial Monitor, the Splash Screen should appear with the description of the parameters in use and immediately start reading the conductivity and temperature, the values shown must be converted as described in the article
63) after connection of all cables, with the hair dryer flow hot air from about 200 mm away on the electrode, the number associated with temperature should vary
64) probably during construction the 2 spirals have been touched with your fingers, so it is recommended to immerse them in acetone for cleaning the surface
65) the conductivity value in the air should be approximately 4030 a.u., now immerse the electrode in tap water, the value should change, all two values, (for Rome it must be around 3100).

Two software lines can be easily changed before compilation

66) *#define between 2775 // delay between measure, as example 3000-(70+155)ms*
fixing the time between two conductivity readings, now 3000 ms (3s), to pass for readings every 10 seconds the new row would be
#define between 9775 // delay between measure, as example 10000-(70+155)ms
without modify the software as a limit 32767

67) *char StringSol[] = "Rome's Tap Water"; // what "solution" we are checking ?*
that produces the output string with text in quotation marks, useful for classifying results, a possible change can be
char StringSol[] = "Turtles Fountain in Rome"; // what "solution" we are checking ?

It is recommended to copy the 2 lines leaving the originals as a comment, for example
// #define between 2775 // delay between measure, as example 3000-(70+155)ms
#define between 9775 // delay between measure, as example 10000-(70+155)ms

// char StringSol[] = "Rome's Tap Water"; // what "solution" we are cheking ?
char StringSol[] = "Turtle's Fountain in Rome"; // what "solution" we are cheking ?

The step n.65 confirms the correct operation of the system (electrode, sketch, software), you now go to your calibration.

Obtain the temperature calibration curves

Start by calibrating the temperature sensor, which at the same time allows you to check that the conductivity sensor is functioning correctly.
68) Carefully wash an ice cube tray, fill it with at least 1 L of distilled water and place it in the freezer until the water has frozen.
69) Fix the thermometer bulb with a rubber band next to the MCP9701A without covering the bulb and the sensor, see Fig. G.
70) Wash thoroughly with distilled water.

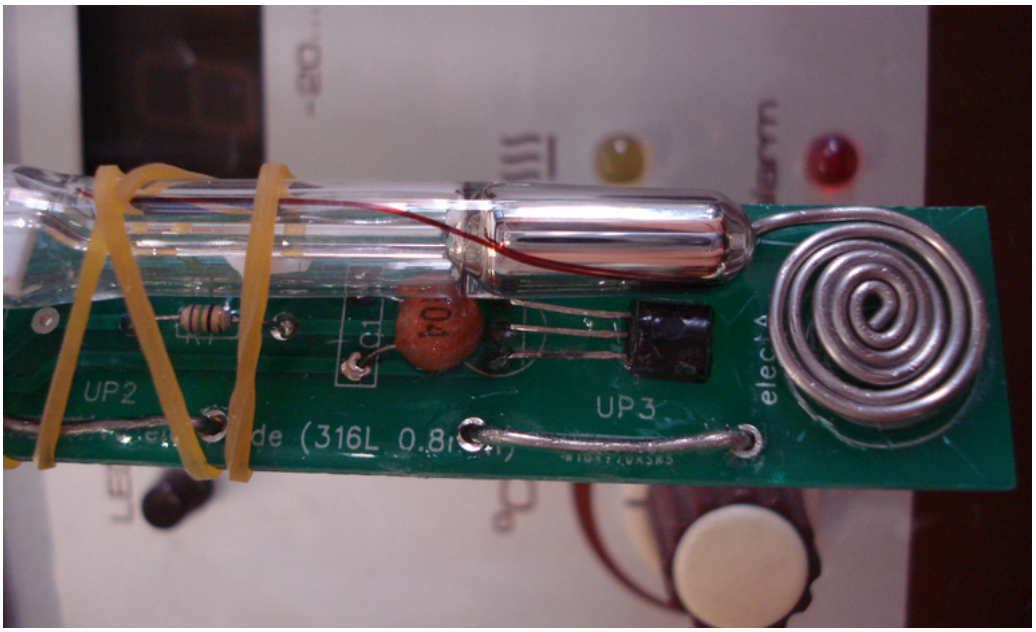


Fig.G. Hg thermometer and PCB electrodes fixed before immersion

71) Fill a carefully washed container (glass, stainless steel) with 1 L of distilled water.
72) Insulate the container by wrapping it with one or two rounds of rolled cardboard, secured with rubber bands, and with a plastic lid with a large hole in the centre
73) place the container on a cooker and, through the central hole in the lid, insert the electrode + thermometer up to 20 - 40 mm from the bottom of the container.
74) Make the electrode - board - Arduino connections as shown in figure 1 of the paper (remember that the temperature sensor does not pass through the board but goes directly to the Arduino). Start the IDE and turn on the *serial monitor*; the file *in-Air-and-Tap-Water.txt* in the SM shows an example of what you should get.
75) at about 25 ˚C, values greater than 4000 and 1500 should appear for conductivity and temperature respectively; if these values are obtained continue with the next step
76) lift the lid, pour all the ice into the beaker and add distilled water almost to the rim, cover again with the lid
77) wait until the mercury thermometer reads about 0 ˚C, write down the temperature value on a notebook together with the order number; on the Arduino serial-monitor will appear at each order number the corresponding signals read out for Temperature and conductivity
78) follow the slow increase in temperature due to heat exchange with the environment and mark the data corresponding to successive changes in 0.5 ˚C steps
79) when the rise in temperature becomes too slow, heat the container by placing it on a hotplate; adjust its temperature as you go along to achieve sufficiently slow growth to allow readings to be taken every 0.5 ˚C
80) the software used does not allow readings above 35 degrees for the MCP9701a, see the final section with possible modifications; when this temperature is reached highlight all the text listed in the serial-monitor, copy and paste it into a text management software, e.g. Windows Notepad
81) save the notepad file with a suitable filename
82) Import the file in a spreadsheet (Excel, Calc, etc.) so to get 4 columns as in the following example

measure conductivity temperature

(num)	(a.u.)	(a.u.)		
1	2494	1416		
2	2499	1390		
3	2499	1424		
3	2497.33	1410.00	avg	
4	2500	1387		
5	2501	1409		
6	2501	1450		
6	2500.67	1415.33	avg	
7	2501	1372		
8	2502	1401		
9	2503	1392		
9	2502.00	1388.33	avg	1.5
10	2503	1447		
11	2504	1378		
12	2504	1413		
12	2503.67	1412.67	avg	

83) using the notebook values add a column with the temperature read by the thermometer to the corresponding measurement number, only as an example see the value in the table highlighted in green
84) build the calibration graph and find the equation that fit it at the best, see data used to obtain the fig. H in the file *Temperature-Calib-1.2.XLS* in SM.
85) If your data are similar to those in the example, the equation obtained can be used to obtain the temperature of the solution in which the electrode is immersed from the bit value (a.u.) coming from the Arduino.

The trend obtained for the conductivity measured by the electrode (green line, non-linear), fig. H, is congruent with the temperature variation, so it can be considered a preliminary test of the correct functioning of the whole instrument.

Conductivity calibration curve

The unusual shape of the electrode, the possible difference of the two spirals and the design of the circuit resulted (as in many instruments) in a non-linear response; thus, the calibration described on p.28, 2.39 of BIPM {3} cannot be performed. The calibration graph will then be obtained using an indirect method, known as the "Secondary Measurement Standard" (see procedure on p. 48, 5.5 note 2 {3}) which requires at least 11 different real samples to be used.

The method also requires a conventional laboratory conductivity meter with an upper

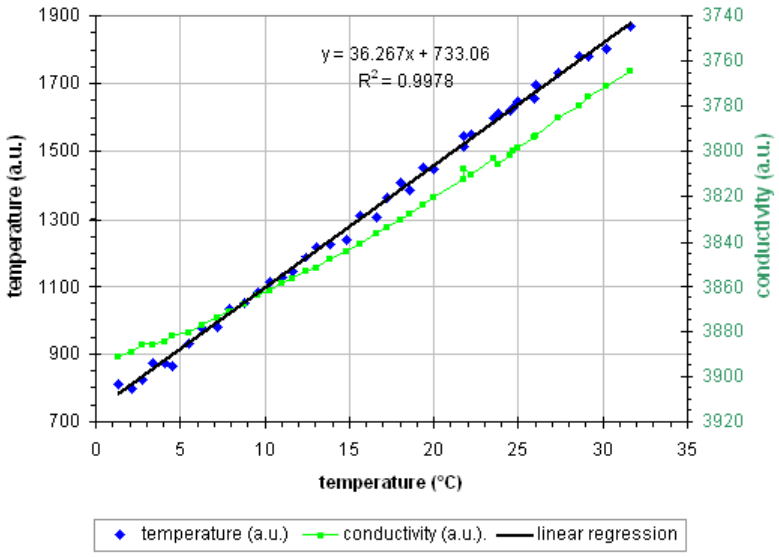


Fig.H, temperature calibration, the corresponding conductivity also shown

limit of conductivity of 1999 uS, i.e. comparable to that corresponding to the maximum response of the Arduino with the values chosen for R1V and R2V.

As an example of sample selection we can take 10 commercial mineral waters (not carbonated at origin and without added CO2) covering from 20 uS up to 2000 uS, adding the distilled water and 2 standards we get 13 samples. There is no limit to the number of samples, the curve improves as they increase if their conductivity values are well spaced.

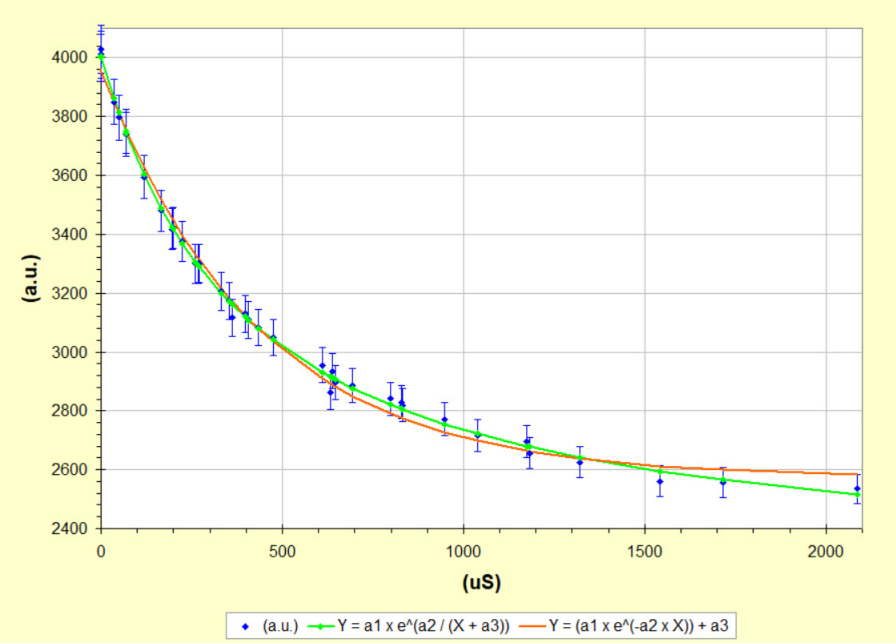
86) before start the calibration all the samples (bottled waters, distilled water and standards) must be left on the measurement bench until the same temperature is reached
87) For maximum reproducibility it is necessary to place both the measuring cell of the prototype and the measuring cell used with the laboratory instrument (Reference Conductimeter, CfR) at the same level for all measurements. For this purpose, cut a 50 mL Falcon to approximately 25 mL, in such a way an optimal geometry for the immersion of the two cells is also obtained.
88) The CfR should be calibrated as described in its manual using the standards at room temperature .
89) connect the Arduino to the computer with the software already installed
90) the Serial Monitor should show the output data
91) calibration starts with the lowest conductivity sample, i.e. distilled water, and continues with those of gradually increasing conductivity
92) For each sample respect the sequence: priming (immerse both conductivity cells in the sample); measurement with the CfR; measurement with the prototype. At the end of each series of operations wash both the Falcon and the conductivity cells thoroughly and dry everything as well as possible.
93) After each measurement with the CfR, mark the conductivity value in the laboratory notebook.
94) After each measurement with the prototype, mark the average conductivity and temperature values, reported on the serial monitor of the Arduino IDE, on the lab notebook
95) put all data from the notebook into a spreadsheet

As an example, the following table lists the values obtained for all samples utilized with the prototype described in the paper. The number of samples is well over the minimum number required (38 samples) and made it possible to obtain a very detailed calibration curve (see fig. I).

brand at label	calibration value (uS)	Arduino value (a.u.)	brand at label	calibration value (uS)	Arduino value (a.u.)
in air	0.0	4028.8	Clivia	407.2	3109.8
MilliQ	0.3	4010.2	Tap water	432.7	3084.0
Distilled	1.2	3999.8	standard600	474.1	3049.4
Sant'Anna 2016	35.8	3849.0	Ferrarelle	610.9	2955.4
S. Bernardo	48.7	3796.8	Carrefour Ofelia	633.6	2863.2
Valmora	67.8	3749.4	Nepi 2018	638.6	2934.8
standard84	69.6	3739.4	Grazia	647.3	2894.8
Levissima	117.6	3595.0	Santagata	694.3	2885.0
Fiuggi	165.1	3480.8	Sangemini	799.9	2841.0
Rocchetta	196.0	3418.0	Claudia	828.9	2829.7
Santa Croce12	197.6	3422.0	Vivia	832.3	2819.8
Santa Croce17	224.3	3376.4	Uliveto 2019	948.9	2771.8
Nestle Vera	260.9	3299.6	Uliveto 2015	1038.9	2717.2
Tullia	267.0	3299.0	Sveva	1174.4	2695.8

Santa Vittoria	269.8	3300.8	standard1413	1181.9	2656.0
Perla	332.2	3204.8	Gaudianello	1321.2	2625.4
Lete	353.7	3173.8	4Gau+1Ess	1541.9	2561.4
Natia	362.1	3116.2	1Gau+1Ess	1716.4	2555.2
Sorgesana	397.3	3129.6	Essenziale	2086.8	2534.6

The trend of the curve is well fitted by two negative exponential equations, that are better described in the paper, each one give best results in a specific conductivity range



To obtain the 2 equations you can use one of the non-linear fitting software, or a statistical programming software such as Matlab, Mathematica, or an online calculator, as an alternative use the software, in SM, "*Electrode-Calibration.XLS*" replacing the values in the spreadsheet appropriate columns with your experimental data.

The above mentioned software, following the instructions of the paper, using the Solver module {4}, calculates the values of the coefficients of 2 different exponential curves. One in orange that better follows the low concentration values of salts and one in green for the more concentrated solutions.

Choose a calibration curve (as an example Y=a1*e^(a2 / (X + a3)) and the calculate coefficients a1, a2, a3.

Just substitute in X the value of a.u. measured for an unknown sample to obtain the its conductivity value Y.

You could enter both the coefficients and the equation in the Arduino firmware but this should be changed every time you change or modify the electrode and/or the values of the components of the shield. The same could be done for the temperature calibration curve.

To modify the software, a guide to Arduino, in Italian {5}, and one to its use in environmental measures {6}, may be useful,

Modify the software

Apart from inserting the equations and coefficients directly into the software as already said, it is therefore advisable to first read an Arduino manual {7} and also a programming guide in C ++ {8}.

For the temperature measurement, a one-bit oversampling technique is used to obtain a higher resolution as from an 11-bit ADC converter, thus obtaining values between 0 and 2047 a.u..

To further increase the resolution, a function of the uC (microcontroller Atmel, ATmega328) is used which allows to set the reference voltage, AREF, of the ADC with the instruction in C *analogReference (... ..)*;

In the firmware it is set to 1.1V but in this way it is possible to read output values from the MCP9701A up to 1.1V, that is to say about 35 º. A modification can be set the AREF to 3.3V using a bridge to the appropriate pin, or to 5V with the command in C ++ *analogReference (DEFAULT)*;

Even the reading of the potential produced by the electrodes uses a 12-bit oversampling technique to obtain values between 0 and 4095 and a higher resolution, even in this case you can make changes to the software.

Modification is possible but not trivial, we recommend reading all the suggested texts {5, 6, 7, 8}, more competence in C++ may be useful,

Reference

1) Andrew Miller, Arduino for Beginner, Rev. 2, Makerspaces Ed., 2017, https://www.makerspaces.com/wp-content/uploads/2017/02/Arduino-For-Beginners-REV2.pdf

2) M. Banzi, Getting Started with Arduino, 2nd edition, Oreilly Ed., 2011, ISBN:978-1-449-30987-9, at: http://phylab.fudan.edu.cn/lib/exe/fetch.php?media=yuandi:arduino:getting_started_with_arduino_v2.pdf

3) BIPM, Vocabulaire international de métrologie – Concepts fondamentaux et généraux et termes associés (VIM), JCGM/WG 2 Ed., 2008

4) Jonathan P. Pinder, An Excel Solver Exercise to Introduce Nonlinear Regression, Decision *Sciences Journal of Innovative Education*, Volume 11, Number 3, July 2013, pp263-278.

5) P. Aliverti, Il manuale di Arduino guida completa, Zeppelinmaker Ed., 2015 at: http://www.zeppelinmaker.it/files/Arduino-Manuale-v0.5.pdf

6) R. Barberi, Arduino misurare e controllare, Univ. della Calabria, Rende, IT, 2014 at: http://www.fis.unical.it/files/fl178/8774arduinobarberilow.pdf

7) J. Borchers , Arduino in a Nutshell, 2015, at: hci.rwth-aachen.de/arduino

8) J. Purdum, Beginning C for Arduino, 2nd ed. Apress Ed., 2015, ISBN: 978-1-4842-0941-7 at: http://ndl.ethernet.edu.et/bitstream/123456789/26653/1/Jack%20Purdum.pdf